Министерство образования и науки Российской федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского»

Институт Информационных Технологий Математики и Механики

Отчет

по учебной практике

**Арифметические операции с полиномами**

Выполнил:

 студент гр. 381703-1

Громов Н.В.

Проверил:

ассистент кафедры

МОСТ ИИТММ

 Волокитин В.Д.

г. Нижний Новгород

2018г

Оглавление

[Введение 3](#_Toc530768225)

[Постановка учебно-практической задачи 4](#_Toc530768226)

[Руководство пользователя 5](#_Toc530768227)

[Руководство программиста 6](#_Toc530768228)

[Описание структур данных 8](#_Toc530768229)

[Заключение 12](#_Toc530768230)

[Список литературы 13](#_Toc530768231)

[Приложения 14](#_Toc530768232)

# Введение

Изучение полиномиальных уравнений и их решений составляло едва ли не главный объект «классической алгебры».

Техническая простота вычислений, связанных с многочленами, по сравнению с более сложными классами функций, а также тот факт, что множество многочленов плотно в пространстве непрерывных функций на компактных подмножествах евклидова пространства, способствовали развитию методов разложения в ряды и полиномиальной интерполяции в математическом анализе.

Многочлены также играют ключевую роль в алгебраической геометрии, объектом которой являются множества, определённые как решения систем многочленов.  
Особые свойства преобразования коэффициентов при умножении многочленов используются в алгебраической геометрии, алгебре, теории узлов и других разделах математики для кодирования или выражения многочленами свойств различных объектов.

# Постановка учебно-практической задачи

**Цель работы**

Разработать программу, выполняющую арифметические операции с полиномами трех переменных (x, y и z): сложение, вычитание, умножение на константу, умножение двух полиномов. Считается, что полином составлен из мономов от трех переменных с ограничением на степень каждой переменой от 0 до 9. Коэффициенты полинома - вещественные числа. Работоспособность программы необходимо проверить с помощью Google Test-ов. Кроме того, необходимо разработать пользовательское консольное приложение.

**Последовательность выполнения работы**

1. Разработка класса List.
2. Разработка класса Pol, реализующего операции с полиномами.
3. Проверка работы методов классов при помощи Google Test-ов.
4. Создание консольного интерфейса пользователя.

**Исходные данные:**

N – количество мономов в данном полиноме.

P – трехзначное число, степени переменных x,y и z соответственно.

С – константа монома.

**Требуемый результат:**

Answer – полином с выполненной операцией.

**Пример:**



# Руководство пользователя

Для начала работы с программой необходимо открыть файл main.exe.После запуска приложения на экране появится окно. Появится просьба ввода первого полинома(количество мономов, степени и коэффициенты) (см. рис. 1)

.

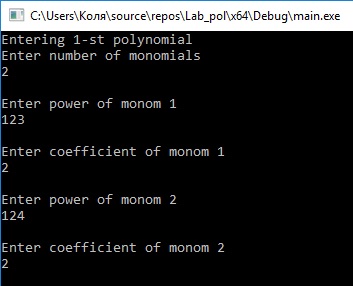


рис.1.Ввод полинома

Аналогично вводится второй полином и после этого программа сообщает об успешном вводе полиномов и показывает доступные с ними операции (см. рис. 2).

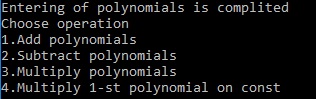


рис. 2. Доступные операции.

После выбора операции на экран выводится ответ(см. рис. 3).



рис.3.Ответ.

# Руководство программиста

Решение поставленной задачи осуществляется с помощью следующих функций, структур и классов.

**Структура Node(struct Node)**

**Класс список(class List):**

**Конструктор по умолчанию(List( )).**

Создает список с указателем головы списка на NULL.

**Функция проверки на пустоту(bool Isempty( )).**

**Функция вставки в конец(void Insert\_end(int val1,double val2)).**

**Функция вставки в начало(void Insert\_begin(int val1,double val2)).**

**Функция удаления элемента списка со значением k(void Del(int k))**

**Функция печати элемента списка с номером k(void Print(int k)).**

**Перегрузка оператора присваивания (List operator=(List a)).**

**Класс полинома(class pol):**

**Конструктор по умолчанию(pol( )).**

Ставит в поле список с указателем головы списка на NULL.

**Конструктор от списка(pol(List b).**

Ставит в поле список b.

**Конструктор от степени и коэффициента монома(pol(int pow, double c)).**

Использует для поля списка функцию вставки в начало.

**Перегрузка операции присваивания(pol operator=(pol b)).**

**Функция умножения полинома на число(pol& Mull(double k)).**

**Функция удаления мономов с нулевыми коэффициентами(void DelZero( )).**

**Перегрузка операции сложения(pol& operator+(pol b)).**

Складывает полиномы путем слияния(с одинаковыми степенями складывает коэффициенты. Остальные мономы второго полинома приписываются в конец полинома результата.). Удаляет мономы с нулевыми коэффициентами.

**Перегрузка операции умножения(pol& operator\*(pol b)).**

Умножает полиномы по правилу произведения полиномов. Удаляет мономы с нулевыми коэффициентами.

**Перегрузка операции вычитания(pol& operator-(pol b)).**

Использует перегрузку операции сложения и умножения на константу(-1).

# 

# Описание структур данных

**Новые структуры:**

**stuct Node**(содержит степень, коэффициент и ссылку на объект типа Node).

**class** **List** (список).

**class pol** (класс полиномов).

**Переменные:**

int k; - переменная, в которую пользователь записывает количество мономов в полиномах, а после этого записывает номер операции из меню.

int p; - переменная, хранящая степень монома.

double c; - переменная, хранящая коэффициент монома.

List a,b; - списки для полиномов.

pol b1,b2,b3; - полиномы(два вводимых и результирующий).

**Общее описание структуры программного комплекса.**

Решение поставленной задачи осуществляется с помощью многофайловой программы.

**Заголовочный файл pol.h**

В данном файле реализован следующие структуры, классы и методы:

**stuct Node**

**Поля:**

double c - коэффициент монома

int p - степень монома

Node\* next - указатель на структуру

**class List**

**Поля:**

Node\* head - голова списка

Node\* tail - хвост списка

int n - количество элементов списка

**Методы:**

**Конструктор по умолчанию(List( )).**

**Функция проверки на пустоту(bool Isempty( )).**

**Функция вставки в конец(void Insert\_end(int val1,double val2)).**

val1 – значение степени

val2 – значение коэффициента

Node\*c – переменная для прохода по списку

**Функция вставки в начало(void Insert\_begin(int val1,double val2)).**

val1 – значение степени

val2 – значение коэффициента

Node\*tmp – переменная для записи новой головы

**Функция удаления элемента списка со значением k(void Del(int k))**

int k – степень удаляемого монома

Node\*c – переменная для прохода по списку

**Функция печати элемента списка с номером k(void Print(int k)).**

int k – степень удаляемого монома

Node\*c – переменная для прохода по списку

**Перегрузка оператора присваивания (List operator=(List a)).**

List a – список, к которому присваивается текущий.

**class** **pol**

**Поля**:

List a – список полиномов.

**Методы:**

**Конструктор по умолчанию(pol( )).**

**Конструктор от списка(pol(List b).**

List b – присваиваемый полю список

**Конструктор от степени и коэффициента монома(pol(int pow, double c)).**

pow – значение степени

c – значение коэффициента

**Перегрузка операции присваивания(pol operator=(pol b)).**

pol b – полином, к которому присваивается текущий.

**Функция умножения полинома на число(pol& Mull(double k)).**

double k – число

pol b – результат

**Функция удаления мономов с нулевыми коэффициентами(void DelZero( )).**

int k – счетчик нолей

Node\*tmp – переменная для внешнего прохода по списку

Node\*tmp1 – переменная для внутреннего прохода по списку

**Перегрузка операции сложения(pol& operator+(pol b)).**

pol b – полином, с которым складывают

Node\*tmp – переменная для прохода по списку текущего полинома

Node\*tmp1 – переменная для прохода по списку из полинома b

**Перегрузка операции умножения(pol& operator\*(pol b)).**

pol b – полином, с которым умножают

pol c – результат

Node\*tmp – переменная для прохода по списку текущего полинома

Node\*tmp1 – переменная для прохода по списку из полинома b

**Перегрузка операции вычитания(pol& operator-(pol b)).**

pol b – полином, который вычитают

**Файл main\_arithmetic.cpp**

Реализован диалог с пользователем. Переменные описаны в структуре данных.

# Заключение

В данной лабораторной работе был изучен ряд известных алгоритмов, и был создан программный комплекс, реализующий такие функциональные алгоритмы, как: алгоритм хранения списком, класс полином и арифметические действия над полиномами. Так же реализован набор гугл тестов, проверяющих правильность программы.

# Список литературы

Кормен, Т., Лейзерсон, Ч., Ривест, Р., Штайн, К. Quicksort // Алгоритмы: построение и анализ / Под ред. И. В. Красикова. 3-е изд., Пер. с англ. —М.: Издательский дом «Вильямс», 2013.

# Приложения

## Приложение 1. Файл pol.h

#ifndef UNTITLED\_LIST\_H

#define UNTITLED\_LIST\_H

#include <iostream>

using namespace std;

struct Node

{

double c;

int p;

Node\* next;

};

class List

{

public:

Node\* head;

Node\* tail;

int n;

List()

{

n = 0;

head = NULL;

tail = head;

}

bool Isempty()

{

return(head == NULL);

}

void Insert\_end(int val1, double val2)

{

n++;

Node\* c = new Node();

c->p = val1;

c->c = val2;

c->next = NULL;

if (Isempty())

head = c;

else

tail->next = c;

tail = c;

}

void Insert\_begin(int val1,double val2)

{

n++;

Node\* tmp = new Node();

tmp->p = val1;

tmp->c = val2;

tmp->next = head;

if (head == NULL)

tail = tmp;

head = tmp;

}

void Del(int k)

{

Node\* c = head;

if (head == nullptr)

return;

if (head->p == k)

{

head = head->next;

n--;

return;

}

while (c->next != NULL)

{

if (c->next->p == k)

{

c->next = c->next->next;

n--;

break;

}

c = c->next;

}

c = head;

while ((c != NULL)&&(c->next != NULL))

c = c->next;

tail = c;

}

void Print(int k)

{

if (k >= n) {

std::cout << "Large number" << std::endl;

return;

}

Node\* c = head;

for (int i = 0; i < k; i++)

c = c->next;

std::cout << c->c <<'('<< c->p<<')';

}

List operator=(List a)

{

head = a.head;

tail = a.tail;

n = a.n;

return(\*this);

}

};

class pol

{

public:

List a;

pol()

{

a.n = 0;

a.head = NULL;

a.tail =a.head;

}

pol(List b)

{

a = b;

}

pol operator=(pol b)

{

a = b.a;

return(\*this);

}

pol(int pow, double c)

{

a.Insert\_begin(pow, c);

}

pol& Mull(double k)

{

pol b;

b = (\*this);

Node\* tmp=b.a.head;

while (tmp->next != NULL)

{

tmp->c \*= k;

tmp = tmp->next;

}

tmp->c \*= k;

b.DelZero();

return(b);

}

void DelZero()

{

int k = 0;

Node\* tmp = a.head;

Node\* tmp1 = tmp;

while (tmp1->next != NULL)

{

if (tmp1->c == 0)

k++;

tmp1 = tmp1->next;

}

if (tmp1->c == 0)

k++;

while ((a.head!=NULL)&&(a.head->c == 0))

{

a.head = a.head->next;

a.n--;

k--;

}

while (k!=0)

{

tmp = a.head;

while((tmp!=NULL)&& (tmp->next != NULL))

{

if (tmp->next->c == 0)

{

a.n--;

k--;

tmp->next = tmp->next->next;

break;

}

tmp = tmp->next;

}

}

tmp = a.head;

while ((tmp != NULL) && (tmp->next != NULL))

tmp = tmp->next;

a.tail = tmp;

}

pol& operator+(pol b)

{

Node\* tmp1 = a.head;

Node\* tmp2 = b.a.head;

while (tmp1->next != NULL)

{

tmp2 = b.a.head;

while (tmp2->next != NULL)

{

if (tmp1->p == tmp2->p)

{

tmp1->c += tmp2->c;

b.a.Del(tmp2->p);

}

tmp2 = tmp2->next;

}

if (tmp1->p == tmp2->p)

{

tmp1->c += tmp2->c;

b.a.Del(tmp2->p);

}

tmp1 = tmp1->next;

}

tmp2 = b.a.head;

while (tmp2->next != NULL)

{

if (tmp1->p == tmp2->p)

{

tmp1->c += tmp2->c;

b.a.Del(tmp2->p);

}

tmp2 = tmp2->next;

}

if (tmp1->p == tmp2->p)

{

tmp1->c += tmp2->c;

b.a.Del(tmp2->p);

}

a.tail->next = b.a.head;

if (b.a.head != NULL)

a.tail = b.a.tail;

a.n += b.a.n;

DelZero();

return(\*this);

}

pol& operator\*(pol b)

{

pol c;

Node\* tmp1=a.head;

Node\* tmp2=b.a.head;

while (tmp1->next != NULL)

{

tmp2 = b.a.head;

while (tmp2->next != NULL)

{

if ((((tmp1->p) / 100 + (tmp2->p) / 100) > 9) || ((((tmp1->p) % 100) / 10 + ((tmp2->p) % 100) / 10) > 9) || (((tmp1->p) % 10 + (tmp2->p) % 10) > 9))

throw "Incorrect power";

c.a.Insert\_end(tmp1->p + tmp2->p, tmp1->c\*tmp2->c);

tmp2 = tmp2->next;

}

if ((((tmp1->p) / 100 + (tmp2->p) / 100) > 9) || ((((tmp1->p) % 100) / 10 + ((tmp2->p) % 100) / 10) > 9) || (((tmp1->p) % 10 + (tmp2->p) % 10) > 9))

throw "Incorrect power";

c.a.Insert\_end(tmp1->p + tmp2->p, tmp1->c\*tmp2->c);

tmp1 = tmp1->next;

}

tmp2 = b.a.head;

while (tmp2->next != NULL)

{

if ((((tmp1->p) / 100 + (tmp2->p) / 100) > 9) || ((((tmp1->p) % 100) / 10 + ((tmp2->p) % 100) / 10) > 9) || (((tmp1->p) % 10 + (tmp2->p) % 10) > 9))

throw "Incorrect power";

c.a.Insert\_end(tmp1->p + tmp2->p, tmp1->c\*tmp2->c);

tmp2 = tmp2->next;

}

if ((((tmp1->p) / 100 + (tmp2->p) / 100) > 9) || ((((tmp1->p) % 100) / 10 + ((tmp2->p) % 100) / 10) > 9) || (((tmp1->p) % 10 + (tmp2->p) % 10) > 9))

throw "Incorrect power";

c.a.Insert\_end(tmp1->p + tmp2->p, tmp1->c\*tmp2->c);

c.DelZero();

return(c);

}

pol& operator-(pol b)

{

pol c;

c = b;

c.Mull(-1);

return(\*this + c);

}

};

#endif UNTITLED\_LIST\_H

## Приложение 2. Файл main.cpp

#include "pol.h"

#include <iostream>

using namespace std;

int main()

{

int k, p;

double c;

List a, b;

cout << "Entering 1-st polynomial" << endl;

cout << "Enter number of monomials" << endl;

cin >> k; cout << endl;

for (int i = 0; i < k; i++)

{

cout << "Enter power of monom" << " " << i + 1 << endl;

cin >> p; cout << endl;

while (p > 999)

{

cout << "Incorrect power, try again" << endl;

cin >> p; cout << endl;

}

cout << "Enter coefficient of monom" << " " << i + 1 << endl;

cin >> c; cout << endl;

a.Insert\_end(p, c);

}

cout << "Entering 2-nd polynomial" << endl;

cout << "Enter number of monomials" << endl;

cin >> k; cout << endl;

for (int i = 0; i < k; i++)

{

cout << "Enter power of monom" << " " << i + 1 << endl;

cin >> p; cout << endl;

while (p > 999)

{

cout << "Incorrect power, try again" << endl;

cin >> p; cout << endl;

}

cout << "Enter coefficient of monom" << " " << i + 1 << endl;

cin >> c; cout << endl;

b.Insert\_end(p, c);

}

cout << "Entering of polynomials is complited";

cout << endl;

cout << "Choose operation" << endl;

cout << "1.Add polynomials" << endl;

cout << "2.Subtract polynomials" << endl;

cout << "3.Multiply polynomials" << endl;

cout << "4.Multiply 1-st polynomial on const" << endl;

pol b1(a);

pol b2(b);

pol b3;

cin >> k; cout << endl;

switch (k)

{

case 1:

{

cout << "(";

for (int i = 0; i < b1.a.n; i++)

{

b1.a.Print(i);

if (i != b1.a.n - 1)

cout << " + ";

}

cout << ")";

cout << " + ";

cout << "(";

for (int i = 0; i < b2.a.n; i++)

{

b2.a.Print(i);

if (i != b2.a.n - 1)

cout << " + ";

}

cout << ")";

cout << " = ";

b3 = b1 + b2;

for (int i = 0; i < b3.a.n; i++)

{

b3.a.Print(i);

if (i != b3.a.n - 1)

cout << " + ";

}

break;

}

case 2:

{

cout << "(";

for (int i = 0; i < b1.a.n; i++)

{

b1.a.Print(i);

if (i != b1.a.n - 1)

cout << " + ";

}

cout << ")";

cout << " - ";

cout << "(";

for (int i = 0; i < b2.a.n; i++)

{

b2.a.Print(i);

if (i != b2.a.n - 1)

cout << " + ";

}

cout << ")";

cout << " = ";

b3 = b1 - b2;

for (int i = 0; i < b3.a.n; i++)

{

b3.a.Print(i);

if (i != b3.a.n - 1)

cout << " + ";

}

break;

}

case 3:

{

cout << "(";

for (int i = 0; i < b1.a.n; i++)

{

b1.a.Print(i);

if (i != b1.a.n - 1)

cout << " + ";

}

cout << ")";

cout << " \* ";

cout << "(";

for (int i = 0; i < b2.a.n; i++)

{

b2.a.Print(i);

if (i != b2.a.n - 1)

cout << " + ";

}

cout << ")";

cout << " = ";

b3 = b1 \* b2;

for (int i = 0; i < b3.a.n; i++)

{

b3.a.Print(i);

if (i != b3.a.n - 1)

cout << " + ";

}

break;

}

case 4:

{

cout << "Enter constant" << endl;

cin >> c; cout << endl;

cout << "(";

for (int i = 0; i < b1.a.n; i++)

{

b1.a.Print(i);

if (i != b1.a.n - 1)

cout << " + ";

}

cout << ")";

cout << " \* " << c << " = ";

b3 = b1.Mull(c);

for (int i = 0; i < b3.a.n; i++)

{

b3.a.Print(i);

if (i != b3.a.n - 1)

cout << " + ";

}

break;

}

default:

break;

}

cin >> k;

return 0;

}